Dijkstra Cheatsheet
(USACO Gold)

Use this algorithm when:

- You are able to figure out step 1 in thinking about the problem
- You are trying to find the shortest path
- The edges have **positive** weights (if they don't have weights use BFS)

Steps to using the algorithm:

Thinking about the problem:

1) Figure out nodes and edges (or states and transitions)
    - A node is basically a state in dp, so for tips on figuring out the node, refer to the tips for figuring out a state in the dp cheatsheet
2) Figure out the start state/node and the end state/node
3) Figure out base cases (also known as bad states/nodes),
    - This usually includes already visited states/nodes
4) Decide if you want to use the O(N^2) or the O((N+M) log N) implementation

Implementation (O((N+M) log N)):

5) Create a priority queue that can store the state/node
6) Add the start state/node to the priority queue
7) Loop for while the priority queue is not empty
    8) If it is the answer state/node, exit the loop
    9) If it is visited, stop don't look at the transitions
    10) Loop through all the transitions/edges
        11) If the new state is not a base case, add it to the priority queue
12) Print the distance to the answer state/node

Implementation (O(N^2):

- This is very similar to the other implementation, the only difference is how you pick which node you look at next. Instead of taking the first item in the priority queue, look through all the nodes and pick the one with the smallest distance

Other Implementation Tips:

- In C++, priority queues are sorted by biggest items first, so make all the distances negative

How to calculate the runtime:

O(N^2) -or- O((N+M) log N)